Internet APIs

Internet Shortcut Shell Extension

Internet Shortcut Shell Extension

Internet Shortcut Shell Extension

Internet Shortcut Shell Extension

Internet Shortcut Shell Extension

Purpose

The Internet Shortcut Shell extension provides a Windows 95 Shell encapsulation of Uniform Resource Locators (URLs). Each URL is stored as an Internet Shortcut, similar to a Shell Shortcut.

Architecture

The Internet Shortcut Shell extension is implemented in a dynamic-link library (DLL), currently called URL.DLL. This library contains code that is generic across the entire class of URLs. URL protocol-specific code is contained in applications or DLLs known as protocol handlers. For example, telnet.exe might contain code to handle URLs of the telnet: protocol.

Internet Shortcuts are typically stored as .URL files in the file system. URL.DLL is the class handler for .URL files. Users can manipulate Internet Shortcuts as they would manipulate Shell Shortcuts. Internet Shortcuts may be created, deleted, opened, mailed, and so on.

URL.DLL implements the InternetShortcut OLE object class. InternetShortcuts expose a number of OLE interfaces to facilitate their manipulation.

User Interface

Creating Internet Shortcuts

The user can create a new Internet Shortcut through the container->New->Shortcut wizard. If a URL is entered in the initial command-line edit control, an Internet Shortcut is created rather than a Shell Shortcut. Some common prefixes are recognized as implying a URL protocol. For example, www.microsoft.com is treated as http://www.microsoft.com. These prefixes are listed in the registry.

Internet Shortcut Property Sheet

Each Internet Shortcut has an Internet Shortcut property sheet associated with it.

The Internet Shortcut property sheet allows the user to modify the Internet Shortcut’s URL, working directory, icon file and index, and show command. Hot key support for Internet Shortcuts may be added.

Implementation

Protocol Handler Association

Protocol handlers are associated with URL protocols via registry entries. The registry entries are similar to those for Shell class handlers for a file class, for example, Notepad’s association with text files. These protocol handler associations may be manipulated by the user through the Windows 95 Explorer->View->Options->File Types property sheet. Internet Shortcuts currently support only the open verb.

ShellExecute() a URL

URLs may also be opened directly by **ShellExecute**(). For example, the user may enter Tray->Start->Run www.microsoft.com to invoke the http protocol handler on the http://www.microsoft.com URL.

OLE Interfaces

InternetShortcuts implement the OLE interfaces listed below.

IPersistFile

IPersistFile may be used to load an InternetShortcut from a file, or save an InternetShortcut to a file. Typically, Internet Shortcut files use the .URL extension.

IShellLink

IShellLink may be used to set and query some of the properties of an InternetShortcut. IShellLink methods may be applied to an InternetShortcut as described below.

SetPath - Sets the InternetShortcut's URL. Same as calling **IUniformResourceLocator::Set**() with no flags set.

GetPath - Copies as much of the InternetShortcut's URL as will fit in the given buffer. Call **IUniformResourceLocator::GetURL**() instead to get the InternetShortcut's full URL.

SetRelativePath - Not implemented.

SetIDList - Not implemented.

GetIDList - Not implemented.

SetDescription - Sets the InternetShortcut's file path.

GetDescription - Gets the InternetShortcut's file path.

SetArguments - Not implemented.

GetArguments - Not implemented.

SetWorkingDirectory - Sets the InternetShortcut's working directory.

GetWorkingDirectory - Gets the InternetShortcut's working directory.

SetHotkey - Not implemented. May be implemented before release.

GetHotkey - Not implemented. May be implemented before release.

SetShowCmd - Sets the InternetShortcut's show command.

GetShowCmd - Gets the InternetShortcut's show command.

SetIconLocation - Sets the InternetShortcut's icon file and index.

GetIconLocation - Gets the InternetShortcut's icon file and index.

Resolve - Does nothing.

For details on IShellLink, see the *Programmer's Guide to Microsoft Windows 95* in the Win32 SDK.

Example Code

To create an InternetShortcut from a URL, use something like this sequence of API calls and methods.

```
#define INC_OLE2    /* for windows.h */
#include <windows.h>
#include <intshcut.h>  /* for Internet Shortcut declarations */

CoCreateInstance(CLSID_InternetShortcut, ...,
   IID_IUniformResourceLocator, ...)
IUniformResourceLocator::SetURL("http://www.foobar.com", 0)
IUniformResourceLocator::QueryInterface(IID_IPersistFile, ...)
IPersistFile::Save(L"foo.url", ...)
IPersistFile::SaveCompleted(L"foo.url")
IPersistFile::Release()
IUniformResourceLocator::Release()
```

Reference

IUniformResourceLocator Interface

Methods for manipulating uniform resource locators (URLs).

See Also

URLAssociationDialog, **TranslateURL**

- **IUniformResourceLocator::GetURL Method**
- **IUniformResourceLocator::InvokeCommand Method**
- **IUniformResourceLocator::SetURL Method**

IUniformResourceLocator::GetURL Method

HRESULT GetURL(PSTR * *ppszURL***)**

Retrieves an object's URL.

Parameters

ppszURL

A pointer to a **PSTR** to be filled in with a pointer to the object's URL. When finished, this string should be freed by calling **SHFree**().

Return Value

Returns one of the following return codes on success:

S_OK

The object's URL was retrieved successfully. **ppszURL* points to the URL string.

S_FALSE

The object does not have a URL associated with it. **ppszURL* is NULL. Otherwise, returns one of the following return codes on error:

E_OUTOFMEMORY

There is not enough memory to complete the operation.

IUniformResourceLocator::InvokeCommand Method

HRESULT InvokeCommand(PURLINVOKECOMMANDINFO *purlici)*

Invokes a command on an object's URL.

Parameters

purlici
: A pointer to a **URLINVOKECOMMANDINFO** structure describing the command to be invoked.

Return Value

Returns one of the following return codes on success:

- S_OK
  - The object's URL was opened successfully.
- S_FALSE
  - The object does not have a URL associated with it. Otherwise, returns one of the following return codes on error:
    - E_OUTOFMEMORY
      - There is not enough memory to complete the operation.
    - IS_E_EXEC_FAILED
      - The URL's protocol handler failed to run.
    - URL_E_INVALID_SYNTAX
      - The URL's syntax is invalid.
    - URL_E_UNREGISTERED_PROTOCOL
      - The URL's protocol does not have a registered protocol handler.

IUniformResourceLocator::SetURL Method

HRESULT SetURL(PCSTR *pcszURL*, **DWORD** *dwInFlags***)**

Sets an object's URL.

Parameters

pcszURL
  The URL to be used by the object.

dwInFlags
  A bit mask of from the **IURL_SETURL_FLAGS** enumeration.

Return Value

Returns one of the following return codes on success:

S_OK
  The object's URL was set successfully. Otherwise, returns one of the following return codes on error:

E_OUTOFMEMORY
  There is not enough memory to complete the operation.

URL_E_INVALID_SYNTAX
  The URL's syntax is invalid.

aStartOfTopic2$Internet Shortcut APIs

TranslateURL

HRESULT TranslateURL(PCSTR *pcszURL*, **DWORD** *dwInFlags*, **PSTR *** *ppszTranslatedURL***)**

Applies common translations to a URL string, creating a new URL string.

Return Value

Returns one of the following return codes on success:

S_OK
: The URL string was translated successfully, and **ppszTranslatedURL* points to the translated URL string.

S_FALSE
: The URL string did not require translation. **ppszTranslatedURL* is NULL. Otherwise, returns one of the following return codes on error:

  E_FLAGS
  : The flag combination passed in *dwInFlags* is invalid.

  E_OUTOFMEMORY
  : There is not enough memory to complete the operation.

  E_POINTER
  : One of the input pointers was invalid.

Parameters

pcszURL
: A pointer to the URL string to be translated.

dwInFlags
: A bit mask of flags from the **TRANSLATEURL_IN_FLAGS** enumeration.

ppszTranslatedURL
: A pointer to the newly created translated URL string, if any. **ppszTranslatedURL* is valid only if S_OK is returned. If valid, **ppszTranslatedURL* should be freed by calling **LocalFree**(). **ppszTranslatedURL* is NULL on error.

Comments

TranslateURL() does not perform any validation on the syntax of the input URL string. A successful return value does not indicate that the input or output URL strings are valid URLs.

URLAssociationDialog

HRESULT URLAssociationDialog(HWND *hwndParent*, **DWORD** *dwInFlags*, **PSTR** *pszAppBuf*, **UINT** *ucAppBufLen***)**

Invokes the unregistered URL protocol dialog box.

Return Value

Returns one of the following return codes on success:

S_OK
: Application registered with URL protocol.

S_FALSE
: Nothing registered. One-time execution via selected application requested. Otherwise, returns one of the following return codes on error:

  E_ABORT
  : The user canceled the operation.

  E_FLAGS
  : The flag combination passed in *dwInFlags* is invalid.

  E_OUTOFMEMORY
  : There is not enough memory to complete the operation.

  E_POINTER
  : One of the input pointers is invalid.

  URL_E_INVALID_SYNTAX
  : The URL's syntax is invalid.

Parameters

hwndParent
: A handle to the window to be used as the parent window of any posted child windows.

dwInFlags
: A bit mask of flags from the **URLASSOCIATIONDIALOG_IN_FLAGS** enumeration.

pszAppBuf
: A buffer to be filled in on success with the path of the application selected by the user. *pszAppBuf*'s buffer is filled in with the empty string on failure.

ucAppBufLen
: The length of *pszAppBuf*'s buffer.

IS_E_EXEC_FAILED constant

const HRESULT IS_E_EXEC_FAILED;

The URL's protocol handler failed to run.

URL_E_INVALID_SYNTAX constant

const HRESULT URL_E_INVALID_SYNTAX;

The URL's syntax is invalid.

URL_E_UNREGISTERED_PROTOCOL constant

const HRESULT URL_E_UNREGISTERED_PROTOCOL;

The URL's protocol does not have a registered protocol handler.

MIME in Windows 95

Description

With the increasing importance of Internet capability in personal computers, it has become more important for Internet-aware applications to share configuration information. MIME content type binding information is stored in the Windows 95 registry where any interested application may manipulate it. Each registered extension may have an associated MIME content type, and each registered MIME content type may have a default associated extension.

Registry Format

```
HKEY_CLASSES_ROOT
  .txt
    <Default> REG_SZ txtfile
    Content Type REG_SZ text/plain
  txtfile
    shell
      open
        command REG_SZ c:\windows\notepad.exe %1
  MIME
    Database
      ContentType
        text/plain
          Extension REG_SZ .txt
          Encoding REG_DWORD 7 (0, 7, or 8 - default 0)
```

A MIME-aware application may request a browser to use for an unregistered MIME content type using the **MIMEAssociationDialog**() API exported by url.dll.

MIMEAssociationDialog

HRESULT MIMEAssociationDialog(HWND *hwndParent*, **DWORD** *dwInFlags*, **PCSTR** *pcszFile*, **PCSTR** *pcszMIMEContentType*, **PSTR** *pszAppBuf*, **UINT** *ucAppBufLen***)**

Invokes the unregistered MIME content type dialog box.

Return Value

Returns one of the following return codes on success:

S_OK
: MIME content type associated with extension. Extension associated as default extension for content type. Application associated with extension.

S_FALSE
: Nothing registered. One-time execution via selected application requested. Otherwise, returns one of the following return codes on error:

  E_ABORT
  : The user canceled the operation.

  E_FLAGS
  : The flag combination passed in *dwInFlags* is invalid.

  E_OUTOFMEMORY
  : There is not enough memory to complete the operation.

  E_POINTER
  : One of the input pointers is invalid.

Parameters

hwndParent
: A handle to the window to be used as the parent window of any posted child windows.

dwInFlags
: A bit mask of flags from the **MIMEASSOCIATIONDIALOG_IN_FLAGS** enumeration.

pcszFile
: A pointer to a string indicating the name of the file containing data of *pcszMIMEContentType*'s content type. Ignored if MIMEASSOCDLG_FL_USE_DEFAULT_NAME is set.

pcszMIMEContentType
: A pointer to a string indicating the content type for which an application is sought.

pszAppBuf
: A buffer to be filled in on success with the path of the application selected by the user. *pszAppBuf*'s buffer is filled in with the empty string on failure.

ucAppBufLen
: The length of *pszAppBuf*'s buffer.

Comments

MIMEAssociationDialog() does not perform any validation on the syntax of the input content type string. A successful return value does not indicate that the input MIME content type string is a valid content type.

MIMEASSOCIATIONDIALOG_IN_FLAGS

```
enum MIMEASSOCIATIONDIALOG_IN_FLAGS {
  TRANSLATEURL_FL_GUESS_PROTOCOL
};
```

MIMEAssociationDialog() input flags.

Members

TRANSLATEURL_FL_GUESS_PROTOCOL
If this member is set, the application selected is to be registered as the handler for files of the given MIME type. If this member is clear, no association is to be registered. An application is registered only if this flag is set and the user indicates that a persistent association is to be made. Registration is possible only if the **MIMEAssociationDialog** parameter *pcszFile* contains an extension.